AFRL-RI-RS-TR-2014-192

DUAL-HIERARCHY GRAPH METHOD FOR OBJECT INDEXING AND RECOGNITION

UNIVERSITY OF MARYLAND

JULY 2014

FINAL TECHNICAL REPORT

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)* JULY 2014	**2. REPORT TYPE** FINAL TECHNICAL REPORT	**3. DATES COVERED** *(From - To)* FEB 2012 – FEB 2014

4. TITLE AND SUBTITLE DUAL-HIERARCHY GRAPH METHOD FOR OBJECT INDEXING AND RECOGNITION	**5a. CONTRACT NUMBER** FA8750-12-C-0117
	5b. GRANT NUMBER N/A
	5c. PROGRAM ELEMENT NUMBER 62305E
6. AUTHOR(S) Isaac Weiss, Fan Yang, Larry Davis	**5d. PROJECT NUMBER** VMRG
	5e. TASK NUMBER 00
	5f. WORK UNIT NUMBER 05
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Institute for Advanced Computer Studies University of Maryland College Park , MD 20742	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RIED 525 Brooks Road Rome NY 13441-4505	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RI
	11. SPONSOR/MONITOR'S REPORT NUMBER AFRL-RI-RS-TR-2014-192

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
UMD built an innovative general purpose, inherently robust system for object representation and recognition. The system is model-based and knowledge-based, unlike most of the current methods which rely on generic statistical inference. This knowledge is intrinsic to the objects themselves, based on geometric and semantic relations among objects. Therefor the system is insensitive to external interferences such as viewpoint changes (pose, scale etc.), illumination changes, occlusion, shadows, sensor noise etc. It also handles variability in the object itself, e.g. articulation or camouflage. All available models are represented in one graph consisting of two independent but interlocking hierarchies. One of these intrinsic hierarchies is a "Level of Abstraction" (LOA) hierarchy, going up from specific to generic objects. The other is based on parts of the object.

15. SUBJECT TERMS
object representation, object recognition, model based recognition, knowledge based recognition

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Patrick K. McCabe
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	31	**19b. TELEPHONE NUMBER** *(Include area code)* N/A

Table of Contents

List of Figures....................ii

1. SUMMARY....................1

2. INTRODUCTION....................2

3. METHODS, ASSUMPTIONS AND PROCEDURES....................3
    3.1. Limitation of Machine Learning....................3
    3.2. The Dual Hierarchy....................5
    3.3. The Graph Representation....................7
    3.4. Viewpoint Invariance and 3D Reconstruction....................10
    3.5. The Recognition Algorithm....................12
    3.6. Probabilities of Nodes....................16
    3.7. Low-level Image Processing....................17
    3.8. Learning and Modeling....................17
    3.9. Algorithm Implementation Details....................18
    3.10. Improved SIFT-based Object Identification System....................19

4. RESULTS AND DISCUSSION....................21
    4.1. Main Accomplishments....................21
    4.2. Goal 1: Object Detection and Recognition....................22
    4.3. Goal 2: Object Identification....................24

5. CONCLUSION....................25

6. REFERENCES....................26

List of Figures

Figure 1: Common changes in images....4

Figure 2: The dual-hierarchy graph. Our object is highlighted....8

Figure 3: Intersections in invariant space....11

Figure 4: Two views intersect in invariant space....12

Figure 5: The traversal algorithm....14

Figure 6: Edge map....15

Figure 7: Line segments....15

Figure 8: View 1. Rectangles, boxes....15

Figure 9: View 2. Same ratios of parallel sides and distances as in View 1....15

Figure 10: HMMWV, two views, similar ratios of parallel sides and distances....16

Figure 11: Hierarchical block structure....20

Figure 12: Our system recognizes these images as projections of the same 3D object....22

Figure 13: M54 truck, two views, similar ratios....23

Figure 14: M35 truck, two views, similar ratios....23

Figure 15: Truck, two views, similar ratios....24

Figure 16: Correct match....24

Figure 17: Incorrect match....25

1. SUMMARY

The goal of the project were twofold:

1. Object detection and recognition: build a general-purpose object recognition system, which is robust with respect to all variable conditions that impede such systems, e.g. viewpoint changes, illumination changes, shadows, paint, camouflage, dents etc. This is a medium term goal.

2. Object identification: given a specific object in an image, such as a "red Toyota pick-up truck with a dent in the left door", determined if such an object was already captured in a given visual database. The object in the image is assumed to be already detected. This is a simpler shorter term goal than the first. Currently the detection in this task is done by a human operator. In the medium term the detection will be done automatically by the general-purpose system developed for the first goal above.

For the first goal above we have built an innovative system which is model-based and knowledge-based, with knowledge derived from analysis of objects and images, unlike many of the current methods which rely on generic statistical inference. This knowledge is intrinsic to the objects themselves, based on geometric and semantic relations among objects. Therefor the system is insensitive to external interferences such as viewpoint changes (pose, scale etc.), illumination changes, occlusion, shadows, sensor noise etc. It also handles variability in the object itself, e.g. articulation or camouflage. We represent all available models in one graph consisting two independent but interlocking hierarchies. One of these intrinsic hierarchies is based on parts, e.g. a truck has a cabin, a trunk, wheels etc. The other hierarchy we call the "Level of Abstraction" (LOA), e.g. a vehicle is more abstract than a truck, a rectangle is more abstract than a door. This enables us to represent and recognize generic objects just as easily as specific ones. A new algorithm for traversing our graph, combining the advantages of both top-down and bottom-up strategies, has been implemented.

For the second goal above we have built another system which is based on the common SIFT method.

2. INTRODUCTION

In recent years the computer vision field has come to be dominated by methods of machine learning (ML) borrowed from the field of statistical inference. These methods do not have much specific understanding of images and rely on extensive training from given examples. These methods are relatively easy to apply and to show some success with, but they do not provide a reliable object recognition solution. The problem is not their immaturity; it is a fundamental limitation due to high dimensionality and non-linearity as we shall see. Much more success has been achieved in sub-domains such as face recognition and license plate reading where specific knowledge has been applied but this has not been generalized. In the following we describe the problems with statistical ML methods. We then present our way of applying general shape analysis to provide a general purpose, inherently robust system for representing and recognizing objects.

The challenges of object recognition stem from the high variability of observed objects. Among the many variables that images contain are viewpoint changes (scale, pose etc.), illumination changes, occlusion, articulation, shadows, camouflage, sensor noise etc. This is in addition to variability in the object itself, before even taking the image, such as when a vehicle is dented or damaged, has added or modified parts, has paint or dirt stains etc.

The common methods of machine learning try to combat the variability by training the system with a large set of known images or "templates". Given such training, the system tries to recognize an unknown object by matching it to the known templates. While this has resulted in some success in some areas, the more general problem of variability has not been overcome. There are simply far too many variables to deal with just by the "brute-force" method of adding more training.

It has been recognized that the way to improve the performance of any ML algorithm is to use domain-specific knowledge, namely specific understanding of the particular domain we work on, which in our case is the domain of images. In other words, we need to put the "understanding" back into image understanding. The success stories in vision involve such knowledge in quite narrow sub-domains. For instance, in face recognition the system knows to look for eyes in specific locations in the face. However, there is no such advance in the larger domain of images. Our system applies knowledge pertaining specifically to the image domain. At the same time this knowledge is general *within* the image domain, being based on intrinsic properties of shapes. This enables our method to combine high-level semantics with low-level image processing in both representation and recognition, and to be independent of the source of the image such as EO, IR, LIDAR, LADAR etc., or even CAD models.

Many previous efforts at understanding images involved using some hierarchy, with a few simple objects at the starting level and increasingly more complex objects at subsequent levels. The idea was to combat variability by recognizing simple objects, with less variability, first and then refine the recognitions with more variables. Many different kinds of hierarchies were used, but they were all one dimensional.

Namely, there was only one parameter changing along the hierarchy, such as scale, number of parts etc. We have observed that such 1-D hierarchies are not sufficient to represent even a small, simple set of objects such as polygons. It is little wonder that these methods were not very successful and largely fell out of favor.

A key innovation of our method is to use a more general, 2-dimensional hierarchical scheme combining two independent hierarchies. This scheme is rich enough to represent a full variety of objects. Moreover, our hierarchies are based on an understanding of the intrinsic geometry of objects and their relations with their parts rather than on a general parameter such as scale. One of the hierarchies is based on parts, e.g. a truck has a cabin, a trunk, wheels etc. The other hierarchy we call the "Level of Abstraction (LOA), e.g. a vehicle is more abstract than a truck, a rectangle is more abstract than a door. This enables us to represent and recognize generic objects just as easily as specific ones. We can recognize a truck even if we don't know the specific type or model, something that the current template-based ML method cannot do.

An abstraction hierarchy is important even in recognizing very specific objects or "fingerprinting". A user wants to know if a specific object that he sees, say "a blue SUV with a dent on the right door", already exists in the database. The object may be there already, but most likely observed from a different viewpoint and with different illumination. Thus a template-based method will not be able to match it or even find the right template to match. Our recognition algorithm will go down the abstraction hierarchy to recognize a generic SUV and then recognize the specific dented variant.

The 2D hierarchy has led to another key innovation, concerning the traversal of the hierarchy. Traditionally, there are two competing strategies: bottom-up and top-down, both having advantages and serious problems. Having two hierarchies makes it possible to do both: we go up in the parts hierarchy and down in the abstraction hierarchy. This combines the advantages and avoids the problems of both strategies. This applies to both recognition and learning.

3. METHODS, ASSUMPTIONS AND PROCEDURES

3.1. Limitation of Machine Learning

As we mentioned above, machine learning relies on template matching[1] and thus suffers from all its limitations. There are far too many variables than can be captured by a training set of templates. Machine learning methods try to compensate for this by introducing some statistical assumptions, or an "inductive bias". Examples of common assumptions are minimal conditional independence (Bayesian networks), neural networks, minimal cross validation error, minimal margin (support vector machines), minimal description length (Occam's razor), Gaussian mixtures (principal component analysis), Markov random fields, k-nearest neighbors, probabilistic graph models and many others[2]. All these assumptions are far too simple to be valid in real world situations. The hope of these methods is that by statistically averaging over a large number of training examples they will achieve convergence to reality. However in most cases there are not enough examples to make it practical.

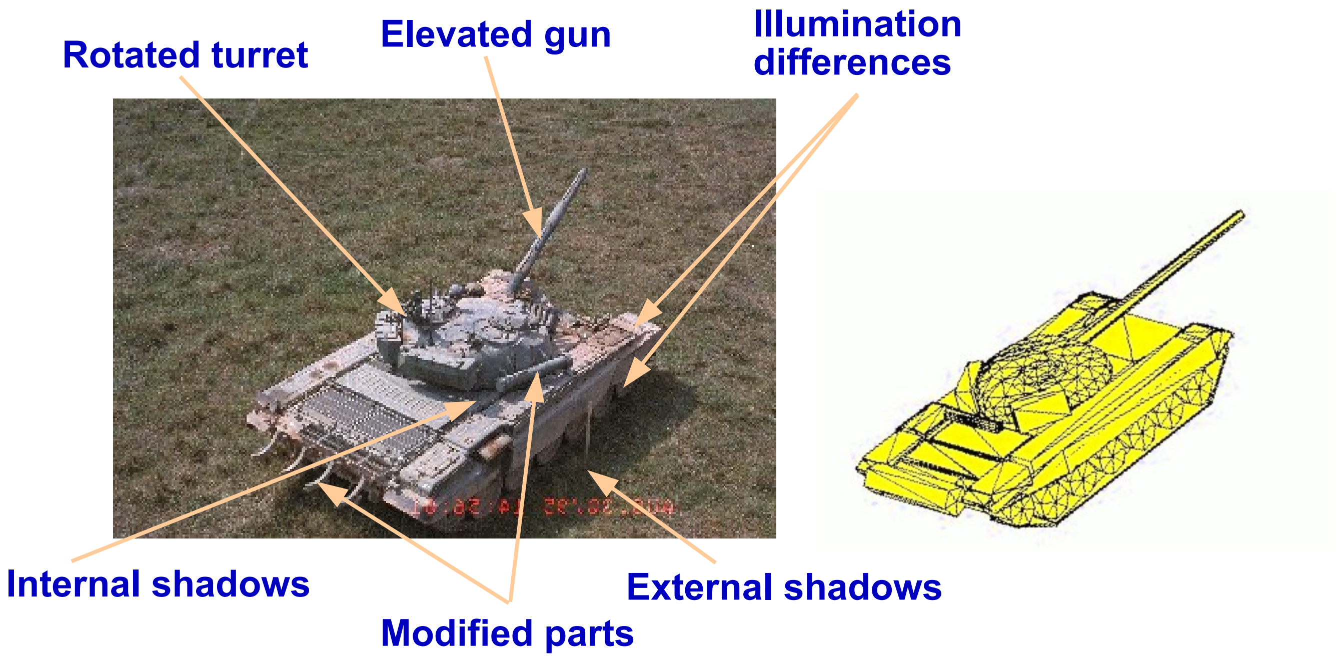

Figure 1: Common changes in images

These statistical assumptions work reasonably well for linear problems of low dimensionality. For example, in line fitting we have the two line parameters that we need to determine by fitting to points in the 2D plane. We assume that any deviation of the points from the line is due to random noise and we average out the noise by assuming a least squares fit (a trivial example of principal components analysis). However, as we have seen, images are influenced by many parameters which are not random noise, from viewpoint variations to shadows etc. This creates a problem of high dimensionality. If we consider each pixel of the image as a dimension, then a small image of 100x100 pixels has 10,000 dimensions. Of course current ML methods use far fewer dimensions but at the cost of loosing important information. Several problems arise here. First, how do we reduce the dimensionality in a judicious way? The automated methods used today do not do a good job of it. Even with reduction we are still left with pretty high dimensionality. Second, how do we populate a high-dimensional space like this with enough data to ensure a meaningful fit? It will require a training set of a size proportional to the volume of an N-dimensional space, namely 10^N, growing exponentially with N. Third, given an image of an object to be recognized, we still have to search for the closest fit in this N-dimensional space, a daunting task. This is often referred to as the “curse of dimensionality”.

Most of the above statistically-based methods use linear methods in dimensionality reduction, assuming that the relevant factors influence the image linearly. However the factors involved in vision are non-linear. Shadows or gray levels depend non-linearly on the positions of several unknown light sources, projection from 3D depends non-linearly on an unknown viewpoint. Thus these generic assumptions cannot work. Other methods involve non-linearities that have nothing to do with vision.

An interesting general result characterizes the performance of ML methods as applied to various problems. It has been encapsulated by a theorem known as the NFL (No Free Lunch) theorem[4], [5]:

Theorem (NFL): Any two search algorithms are equivalent when their performances are averaged over all possible problems.

This means that we are unlikely to improve performance by tweaking some ML algorithm that was applied before or using a variant of it. Over a large set of problems, the performance of all these algorithms will converge to the same value. Given the difficulties of ML in high-dimensionality non-linear problems, this performance value is not very high.

So, how do we "earn" our lunch? What do we have to do to improve performance? It has been recognized that there is only one way to improve performance: use knowledge specific to the problem. Thus we need to use specific knowledge about vision rather than a generic ML assumption. We cannot take the easy route and rely on a statistical machine learning method to do the task of image understanding. We have to use methods of shape analysis to obtain a true understanding of the structure and relationships of images and objects. In fact this is demonstrated by the relative success of vision systems in narrow sub-domains, such as face recognition. These systems rely on specific knowledge about faces such as the expected locations of the eyes. However, the lack of a more general understanding of images makes these systems easily confused by unexpected variations such as sunglasses, different poses etc. In the following we will apply knowledge that pertains to the domain of images in general.

3.2. The Dual Hierarchy

A good way to reduce the dimensionality of images is to use a hierarchy. Many kinds of hierarchies have been tried: scale space, wavelets, "pyramids", quadtrees, Fourier transform, discrete cosine transform, parts decomposition, levels of detail, semantic levels etc. The idea is that the higher levels of the hierarchy have lower dimensionality so recognizing objects becomes much easier. Once an object is recognized at a higher level the information can be used to perform a more refined recognition.

All these methods have one problem in common: they are one-dimensional hierarchies. That is, there is only one parameter that changes as we go up the hierarchy, such as scale or level of detail. However, one dimension is not enough to represent the wide range of possible objects with all their variability. A simple example can demonstrate this. A polygon can be decomposed into parts, namely its sides. The sides are lower in the parts hierarchy. The number of parts can distinguish different polygons from each other, e.g. a triangle, a quadrilateral, or a pentagon. However it cannot distinguish a generic quadrilateral from a rectangle, and a rectangle from a square. These are distinguished from each other by an independent hierarchy, involving specific geometric relations between the parts of the object. A rectangle is more specific than a quadrilateral as its angles are specified as all equal, and a square is more specific than a rectangle as all its sides are equal. We call this hierarchy the "Level of Abstraction", or LOA. We can see that both of these hierarchies, the parts and the LOA, are essential for describing even these simplest of objects.

More practical examples abound. A generic vehicle is composed, in the parts hierarchy, of generic parts such as wheels, a body, doors, windows. A generic wheel is composed of a rim, a tire, a hub and spokes etc. In the Level of Abstraction hierarchy, a generic vehicle can be specified as a car, a truck, an SUV etc. A car can be further specified as a sedan or a sports car etc. Generic wheels can be specified into types by the thickness of the tires, the shape of the spokes etc. Thus the two-dimensional hierarchy of parts and LOA is quite sufficient to represent quite complex examples like these. In the following we will use this 2D hierarchy for our representation of objects.

The abstraction, or specificity, can be defined in several ways. In the example of the rectangle and square, we used the geometric relations between parts, such as their relative sizes or angles. Less specific relations equate to higher level of abstraction. Most of our abstractions are of this nature. However we found another form of abstraction to be useful – a semantic one. For instance, in a CAD model of a car, the roof may be represented as a rectangle. The roof is more specific than the rectangle because it belongs to the car, so it is lower in the LOA hierarchy. In this case the specificity is *semantic* rather than geometric, i.e. the roof has a meaning that a generic rectangle doesn't have. This semantic meaning of a roof comes from the context of the car model rather than from the geometry of the rectangle itself. Having recognized the rectangle as a roof, we can then use knowledge specific to car roofs for further understanding of the car model. In the quadrilateral example, a generic line segment becomes a side of a quadrilateral, a more specific object, through its being a part of the quadrilateral. We can then use known relations between quadrilateral sides to further specify the quadrilateral. This semantic meaning does not derive from human language. It is a "meaning" acquired by a generic object when it becomes part of a bigger object. It then becomes a distinct specific object. This is a form of intersection between the parts and the LOA hierarchies.

Having generic objects, at a higher LOA, is essential if we want to recognize objects that we have not seen before in full specificity. We may have never seen a "Chevy Volt" car, and it may be different in specific details from any car that we have seen before, but we will recognize it as a car because we have the generic object of a car in our database. We will not need specific training. This is hard to achieve with template matching. In other words, we are able to reduce the dimensionality by setting aside the dimensions specific to particular cars. We will describe in the next section how our system represents generic objects at different levels of abstraction.

This ability of the LOA to recognize generic objects is needed even in identifying very specific objects such as in fingerprinting. We want to know if a truck with a dent on the right door has been seen before. We cannot train a system for every possible dent; there may be only one example of the dented truck seen before, and that sample most likely has a different viewpoint, illumination etc. than the object at hand. Thus template matching will not work, and in fact we cannot even efficiently retrieve the right template to match to; we will need to try a large number of templates in the database. Our hierarchical system goes down the LOA hierarchy, finds the generic truck and then retrieves the dented variant.

Both the LOA and the parts hierarchies depend on the intrinsic nature of the object itself rather than on some general parameter such as scale or frequency. Such parameters are not very useful. For instance, a scale space hierarchy is generated by smoothing an image by (e.g.) a Gaussian filter. A limited amount of smoothing may be useful for some noise reduction but when taken too far it totally distorts the shape. A rectangle will loose its corners and gradually morph into a circle, an unrelated shape, as we go up in scale space.

Other kinds of abstractions may be useful but were not used in this project. For example the polygon is higher than both the triangle and the quadrilateral as the number of parts of a polygon is not specified. This may be of use more verbally than visually.

3.3. The Graph Representation

The question immediately arises of how to represent the dual hierarchy of parts and LOA. Parts decomposition is quite easy to represent. However, how do we represent an abstract object such as a generic triangle? This seemingly simple problem has occupied philosophers since Plato. No one has ever seen a generic triangle – all the triangles we have ever seen are specific, having specific values for the lengths of the sides. Yet we do have a concept in our minds of an abstract, generic triangle. Obviously there is a verbal definition of a triangle but we are more interested in a visual representation. Similarly, all the vehicles we have ever seen are specific images or templates, but we do have a visual concept of a generic vehicle. An exact verbal definition of a generic vehicle seems impossible so a more visual representation is essential for such a complex and variable object.

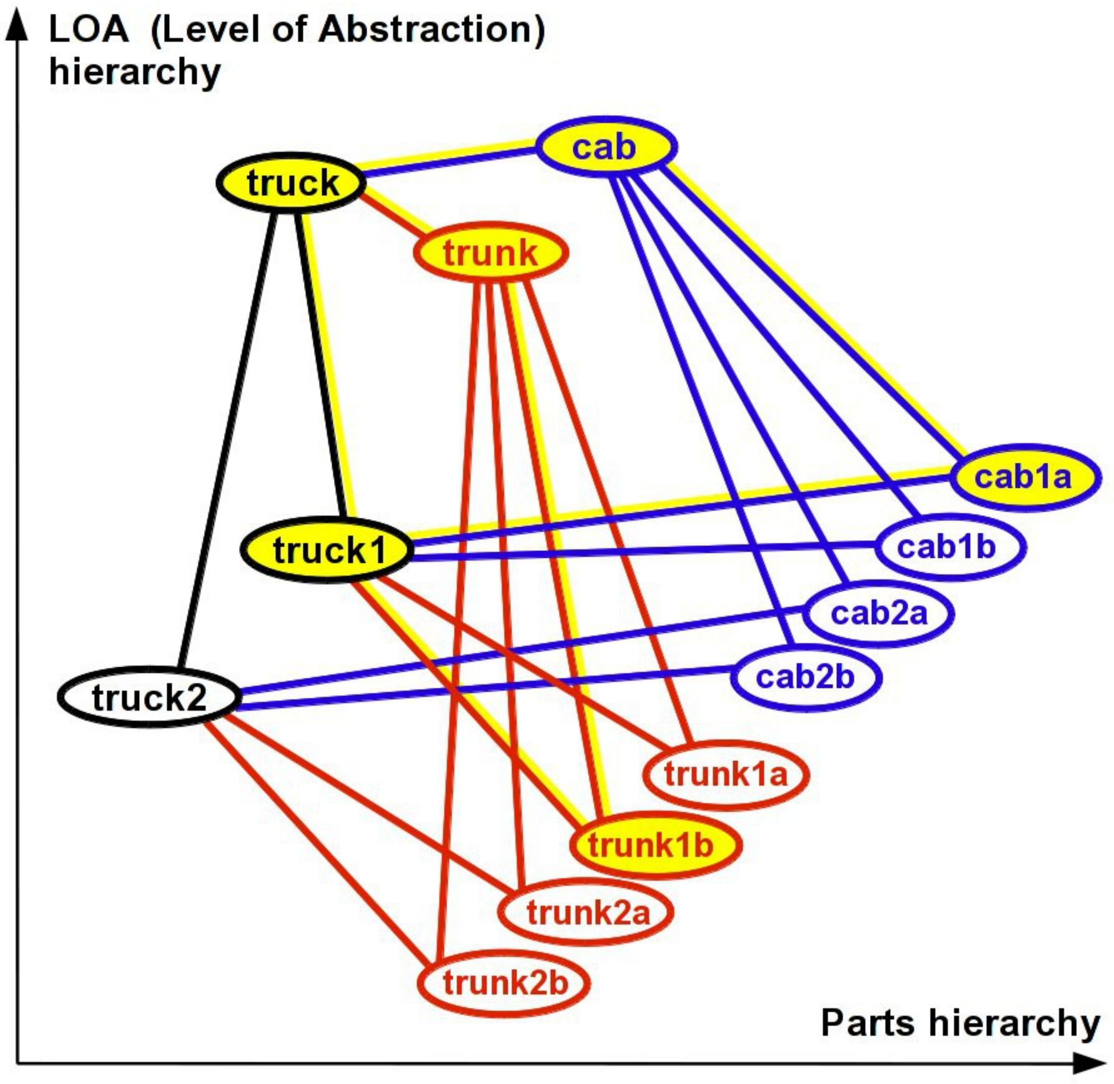

Figure 2: The dual-hierarchy graph. Our object is highlighted.

Our solution is to represent objects as nodes in a graph. This graph is organized as a dual hierarchy that represents both our parts and abstraction hierarchies. All objects, specific and generic, are represented as nodes in the graph. Their relations to each other are represented as links in the graph. A specific instance of an object can also be seen as a small sub-graph that includes the node and its specific parts nodes. Various graph and network methods have been tried before but they were quite amorphous and never had the clear dual hierarchy we use. We do not presume to know if biological vision systems are organized this way but it is easy to imagine that such a graph can be built from neurons.

A simple example is shown in Figure 2. In this figure two trucks and their parts with all known variants are represented in the same graph. The parts hierarchy is shown here left-to-right, i.e. a truck is made up of a cabin and a trunk as seen on each horizontal level. The vertical hierarchy represents the Level of Abstraction (LOA). That is, a truck can be defined generically if we don't have (or don't need) enough detail to identify the specific model, and this is seen at the top level in the LOA hierarchy. Here at the top a generic truck is linked horizontally to a generic cabin and a generic trunk. When we need more specificity, the truck can be recognized as either a ``truck1" or a ``truck2" on a lower level of abstraction. To represent this, our generic truck node is linked vertically to both the "truck1" and the "truck2" nodes lying at the lower LOA level. Similarly, the generic cabin and the generic trunk at the top level are linked vertically to more specific models of cabins and trunks at the lower level. At the same time, on the lower abstraction level, the "truck1" and the "truck2" are linked horizontally (i.e. in the parts hierarchy) to the specific models of their respective parts, namely specific cabins and trunks. A specific instance of truck1 is the sub-graph consisting of the node "truck1", the specific variant parts nodes "cab1a", "trunk1b", and their generic nodes higher up in the LOA. This sub-graph is highlighted in Figure 2.

This example shows that the graph provides a very flexible definition of objects. We are not restricted to one definition of the "truck1" but we accommodate different variants having different variant parts, and we can also place them all as sub-graphs under the generic truck. Thus a potentially large class of specific objects can be organized very efficiently as sub-graphs in the dual hierarchy under the same generic object. At the same time these sub-graphs are as distinct as the objects themselves so matching errors can be minimized.

The graph contains all objects or models known to the system. Each node (object) is connected to other objects on higher and lower levels in both hierarchies. This includes objects from the lowest levels such as edges, corners, or line segments, to mid-level so-called "visual words" based on local appearance descriptors, to the highest levels of objects to be recognized, each object having a position in each of the two hierarchies. In this way we integrate the high level knowledge about structure and (geometry based) classification of objects with low level data.

The levels of semantic abstraction are also represented in the graph. For instance, a circle can be an abstraction of a wheel or of an eye, both having more semantic meanings. Thus the node "circle" is connected to both the "eye" and the "wheel" nodes which are lower in the LOA. The wheel acquires its meaning, e.g., by being a part of a vehicle, while the eye is part of a face. Accordingly, the node "wheel" is also connected to the node "vehicle" which is on a higher level in the parts hierarchy. Thus our graph affords us a very flexible representation of semantic knowledge as defined in Sec. 3.2

In addition to the objects, we need to represent the relations between objects and parts. These are represented by the links between the nodes. The relations can be geometric, e.g. distances, angles, relative sizes of parts etc, or they can be topological, e.g. a part touches another one or contained within it. Topological relations are inherently invariant to geometric transformations such as viewpoint changes. We want our quantitative geometric relations to be invariant too. Distances and angles are not generally invariant, so we will replace them with quantities that are invariant as will be described in the next section. A graph is said to be ``attributed" if the links between the nodes possess some numerical quantities, or ``attributes". Thus our representation consists of an attributed graph containing object nodes and their attributed links.

The graph representation, being intrinsic to the objects, is invariant to external influences such as viewpoint change, illumination, shadows etc. It is also invariant to the platform – whether we use EO, IR, LIDAR, LADAR or a CAD model we can use the same representation. Thus it can be used as a form of a unified index of objects across all platforms as well as semantic levels. The latter are often indexed by separate databases, e.g. a database for raw data, a database for polygon models etc. These semantic levels are closely related to our levels of abstraction which we represent in a unified way in the same graph. We will describe in Section 3.9 how the graph representation is actually implemented on the computer.

3.4. Viewpoint Invariance and 3D Reconstruction

As mentioned before, we want the relations between the parts to be invariant with respect to geometric transformations such as viewpoint changes. Achieving viewpoint invariance also means that we can reconstruct the full 3D object from its projected 2D image, which solves a major problem in image understanding.

The subject of geometric invariants was quite active among mathematicians in the 19th century. However it was unknown in the vision community until a paper by Weiss[6] has surveyed the mathematical literature and introduced it into the field. Since then there have been many applications of invariants in vision research.

Of particular interest is the issue of invariance with respect to the projection from 3D objects to 2D images. Unfortunately the classic literature did not concern itself with this subject but only studied the projection between spaces of the same dimensionality, e.g. 2D to 2D or 3D to 3D. The projection from 3D to 2D was only recently studied. It turns out that there are no true invariants of this projection, but there are invariant *constraints* that are almost as useful. Weiss[8],[10] developed such constraints involving 5 points in 3D space. We will briefly describe it here. Five 3D points posses three invariants, and therefor can be depicted as one point in a 3D space of invariants (not the original 3D space). This point is shown as the red dot in Figure 3. Given a projection onto a 2D image of these 5 points, projected from a certain viewpoint, we can obtain a line in our 3D invariant space. We proved that this line will go through the invariant point (the red dot) generated by the original 5 points (Figure 3). Now given a projection from a second viewpoint, we will obtain a different line but it too will intersect the same invariant point. We have expressed this as a theorem involving determinants [10].

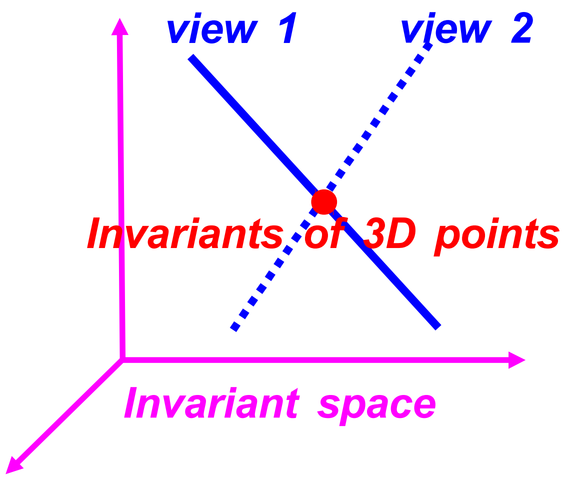

Figure 3: Intersections in invariant space.

We have applied this property to recognizing 3D objects in 2D images. Figure 4 (left) shows 2D images of the same 3D vehicle. After extracting feature points from these images, two sets of lines were generated in the 3D invariant space, shown in Figure 4 (right) in different colors. We checked whether lines from one set intersected the lines from the other set in 3D. If enough such lines intersect then the two images must belong to the same 3D vehicle. The intersection points correspond to the invariant points of the 3D vehicle. Thus we were able to perform recognition without having a 3D model - one view served as a model and the other served as the object to be recognized. The same method was later extended successfully to video sequences in a project carried out in cooperation with SET corporation.

While the method was successful, it had its limitations. In particular, the need to use 5 feature points for every invariant point can lead to a combinatorial problem because there is a large number of possible 5-point sets even with a relatively small number of feature points. This is simply because there are many ways to choose 5 points out of a set of N points, and the number increases rapidly with N. In the current project we do not use points but whole parts. A part has more information in it than a point, e.g. the orientations of its axes, which replaces the need for 5 points. We only need two parts to generate an invariant point which solves the combinatorial problem. This is another advantage of the parts hierarchy beside the invariance of the parts decomposition itself.

In many cases, especially for man-made objects such as vehicles or buildings, we can use simpler invariants. In particular, parallelism is an affine invariant property. If two lines (or line segments) are parallel in 3D, their affine projections into 2D will also be parallel. (Affine projection holds when the object is not too close to the camera). Also, the ratios of lengths of parallel line segments is preserved under this projection. Man-made objects typically contain many parallel lines so these invariants are very useful.

The invariants discussed above are used as attributes of the links between objects/parts instead of quantities such as distances or angles. As the structure of the graph itself is invariant, this makes the whole graph representation invariant to the viewpoint.

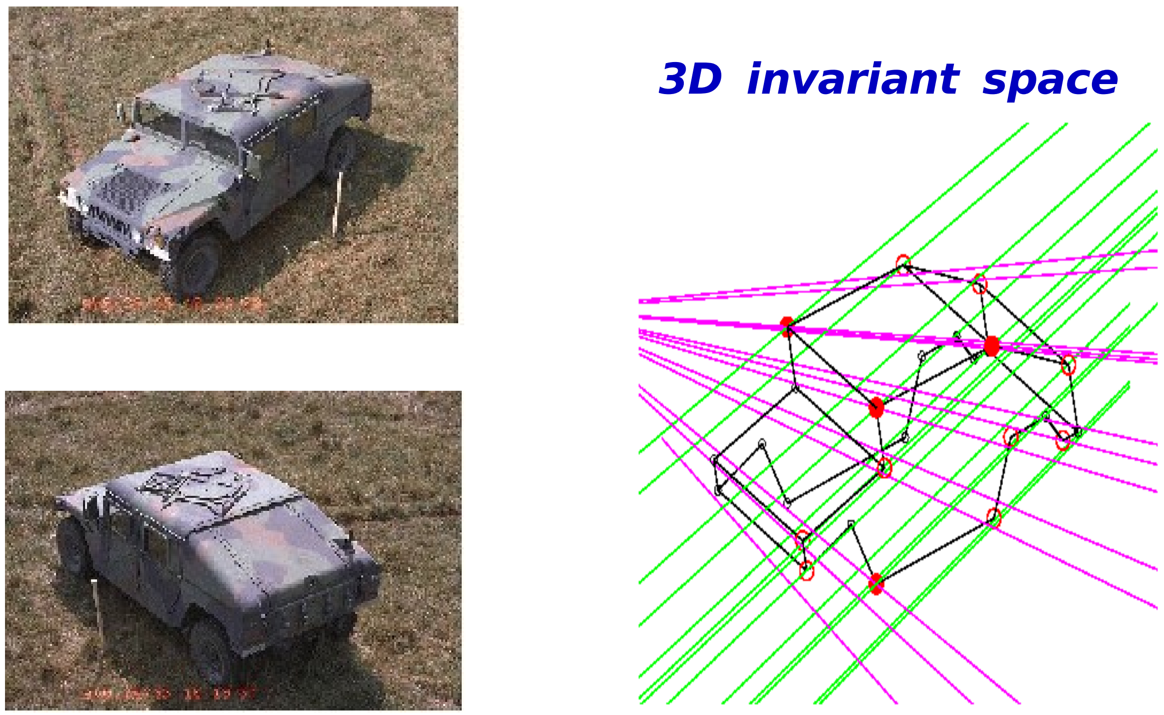

Figure 4: Two views intersect in invariant space.

3.5. The Recognition Algorithm

The graph containing all models is constructed off-line and serves as our database of models. Given an object to recognize, we now need to search for the correct model in the graph hierarchies.

Traditionally there are two competing approaches to traversing such hierarchies: top down and bottom up, each with its own serious problems. Starting at the bottom, there are many raw features such as edges and we face a combinatorial problem of how to group them in the right way to obtain the higher level object. This is greatly compounded by noise and uncertainty in the features. Trying to start from the top, we have to somehow guess which high level object we might have and try to fit it to the low level features. We don't often have such a guess. Part of the problem is the confusion in defining a high- or low-level object. Given a circle, is it a low-level primitive or is it a high level abstraction of a wheel or a face?

Our dual hierarchy makes it possible to use both strategies at the same time, combining their advantages and avoiding their problems. In a nutshell, we use a top down traversal in the level of abstraction hierarchy and bottom up in the parts hierarchy. The circle is both a low-level primitive in the parts hierarchy and a high-level abstraction in the LOA hierarchy. Thus a circle can be a starting point for both hierarchies.

Traditional methods try to segment the image into relevant parts by some heuristics, then group together low level features by more heuristics, and then match the result to the model. This is rarely successful. Our algorithm is directed by the knowledge in the graph rather than by heuristics, namely the grouping and segmentation go hand-in-hand with recognition. We start with a rough tentative grouping of raw data in the image, and then go to the graph and obtain more knowledge about the potential object. Using that we obtain more accurate grouping which leads us to obtaining even more knowledge from the graph. By "obtaining more knowledge" we mean utilizing the links and their attributes as we go down in the abstraction hierarchy towards more specific objects, and up in the parts hierarchy towards higher-level grouping. This traversal proceeds, repeatedly alternating between the parts and LOA hierarchies, until all objects are found.

The algorithm can be illustrated by a simple example shown in Figures 5-9. Nodes are found by a Huygens wave that propagates by means of two functions: a group finder and a parts finder, applied consecutively in each iteration. These are represented by red and green arrows respectively in Figure 5. The starting nodes here are the "line segments" from Figure 7. These are obtained here by a conventional methods such as a Canny edge detector (Figure 6). The algorithm applies the group finder which checks possible relations between line segments, stored in nodes linked to the line segment nodes in the model graph such as the rectangle node. If some of these relations, indicating a projected rectangle, are satisfied then we have recognized a rectangle in 3D. Such relations were satisfied (Figure 8), so nodes "rectangle" are now added to our image graph, Figure 5. There can be many rectangle nodes in the image graph while there is only one in the model graph. Once a rectangle is recognized, we use the parts finder to locate its sides. These are predicted by the model to be in the locations of certain line segments in Figure 7, and so we add the "side" nodes to the graph, below the corresponding line segments nodes in the abstraction hierarchy, with the positions and orientations of these line segments. The nodes we found are highlighted in yellow in Figure 5.

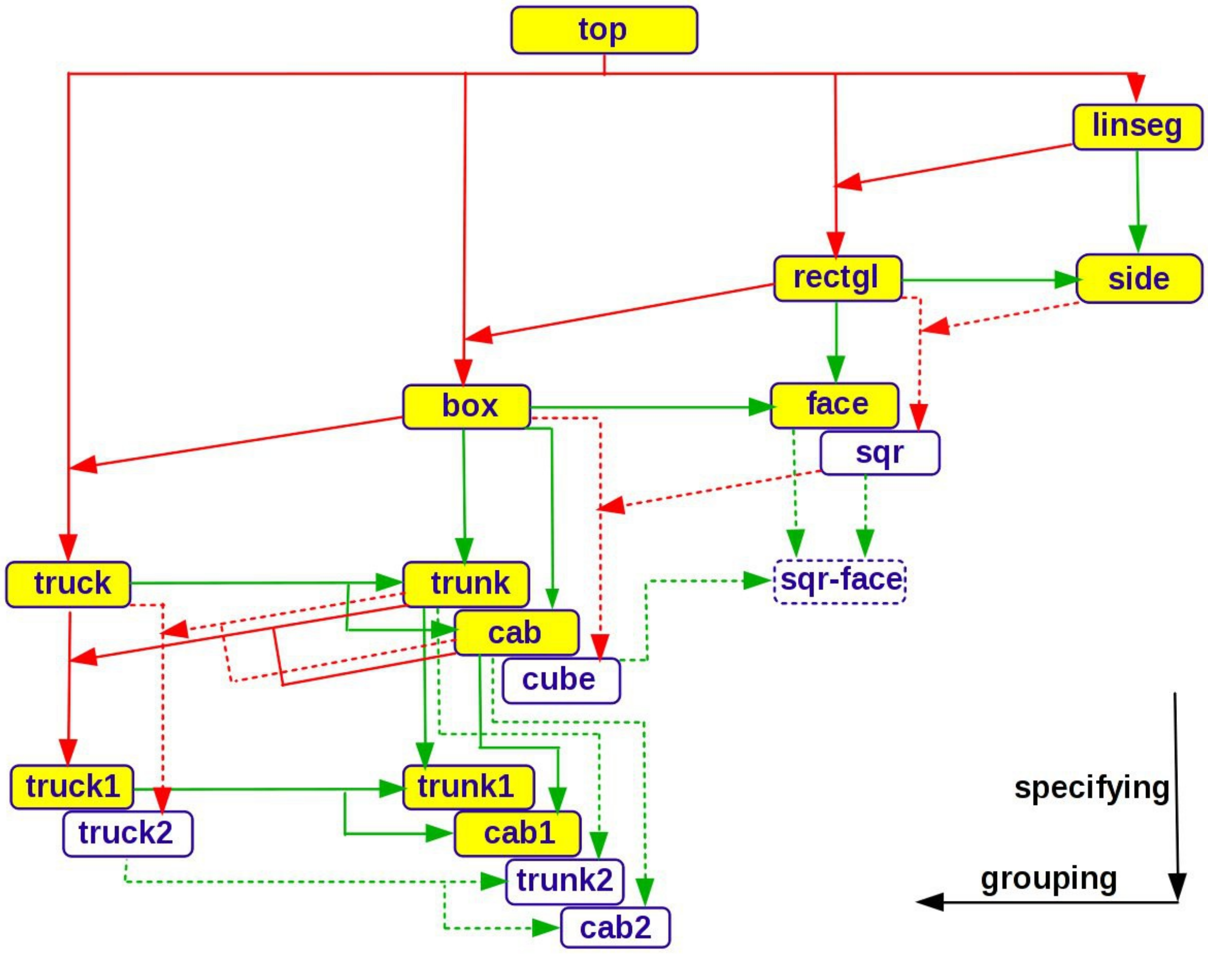

Figure 5: The traversal algorithm.

The next iteration applies the group finder again, this time to the newly discovered nodes, the sides and the rectangles. It first looks for relations between the sides. If two adjacent sides are equal we have recognized a “square”, which is more specific than a rectangle. These relations were not satisfied so no squares were found in this image. (This is represented as dashed lines in Figure 5). The group finder then proceeds to look up 3D invariant relations between rectangles. If certain relations are satisfied, we have found a projection of a “box”, a 3D object represented by a node higher up in the parts hierarchy. The group finder did find boxes so their nodes are added to the image graph. Again we use the parts finder and find the “faces” of the boxes, whose nodes are added under the “rectangle” nodes in Figure 5. In the next iteration the group finder operates on the boxes, checking their 3D invariant relations and finding the generic “truck” in Figure 8. The parts finder then finds the generic “cabin” and “trunk”. These are all added to the image graph in Figure 5.

The parts finder also finds other parts such as wheels and helps to refine the positions and orientations of the parts, at all levels, using the model. It can get rid of noise and shadows because these are not on the list of parts in our model graph of a truck.

In the final iteration the group finder checks more specific relations between a cabin and a trunk than the generic relations in a generic truck. When these relations are satisfied we obtain a more specific “truck1”.

The parts finder finds its specific parts "cabin1" and "trunk1". Their nodes are added. The relations for "truck2" were not satisfied so this node is not added. We show it and its parts nodes in Figure 5 without highlighting.

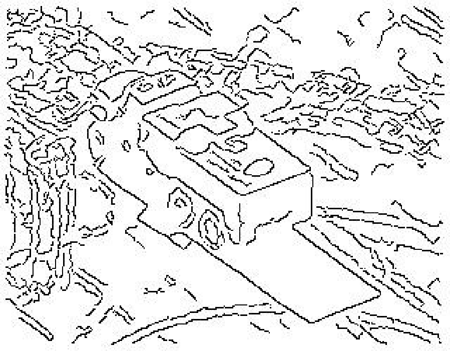
Figure 6: Edge map.

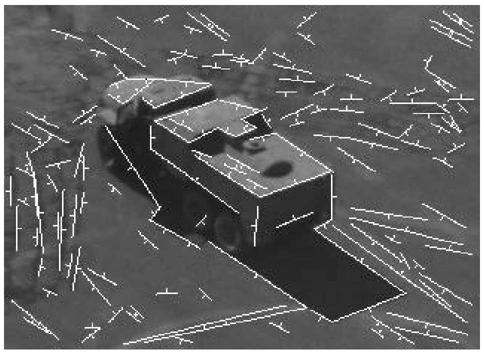
Figure 7: Line segments.

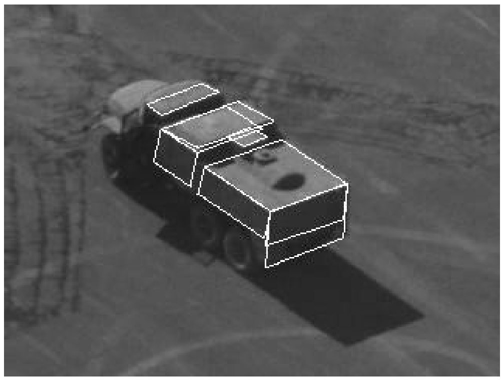
Figure 8: View 1. Rectangles, boxes.

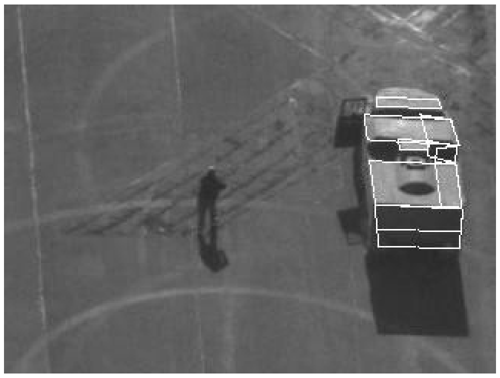
Figure 9: View 2. Same ratios of parallel sides and distances as in View 1.

In summary, we go up in the parts hierarchy and down in the abstraction hierarchy systematically at the same time, simultaneously grouping parts and recognizing more specific objects. This process is guided through all levels of the processing by prior knowledge, expressed in the model graph hierarchies as well as in the invariant relations stored in the nodes.

The method is invariant to various transformations such as viewpoint changes. Figure 9 shows the same algorithm applied to the same truck seen from a different viewpoint. We obtained very similar results for the object's structure and its invariants. That is, the ratios of lengths of parallel sides of various rectangles, as well as the ratios of sides to distances of rectangles in parallel directions, are the same in both views. There is no geometric transformation between two such images projected from 3D, so they cannot be matched by a template matching method even if it tries to account for transformations. Our method

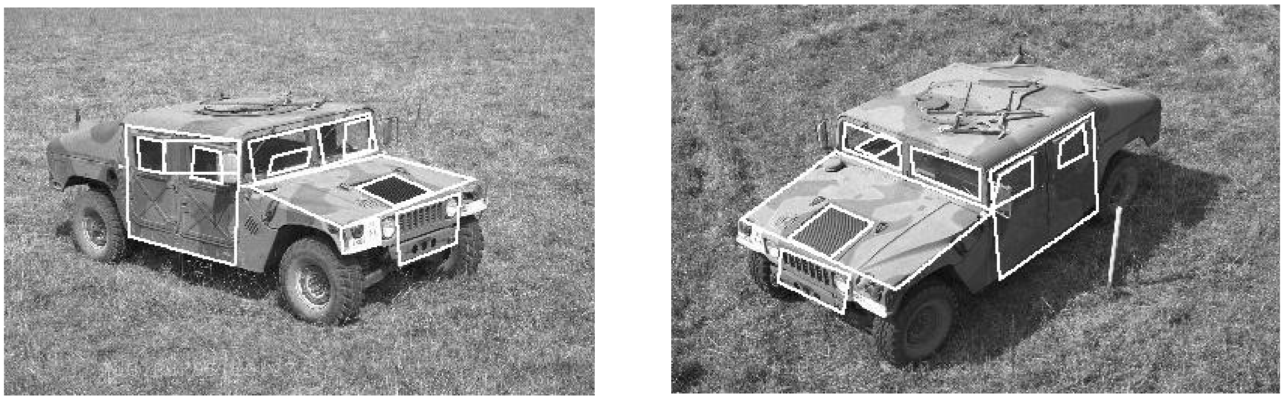

Figure 10: HMMWV, two views, similar ratios of parallel sides and distances.

easily recognizes these two images as projections of the same 3D truck model expressed in the graph. More examples of view independence are shown in Figure 10 and in Section 4.2.

As we have seen, we are able to recognize a generic truck before we recognize the specific one. Again this is unlike template matching or ML.

3.6. Probabilities of Nodes

Of course the actual image of an object does not always correspond perfectly to the model. Thus all our quantitative attributes such as distances or ratios are expressed with elasticity, or "spring" constants that represent the tolerances of these quantities. These yield Bayesian probabilities for the existence of the nodes. Each link of a node from another node in the image graph carries a probability for the existence of this node in the image, conditioned on the other node. This probability is a function of (i) deviations of the positions or orientations of the nodes from the ideal predicted by the model that we fit, (ii) a constant factor set in the model, and (iii) possible influences of adjacent links. The deviations create "potential energies" coming from the elasticity. After each iteration of the algorithm described above we recalculate the energies contributed by all the nodes connected to each node. We take into account nodes both above and below in the two hierarchies. We then adjust the position and orientation of the node by minimizing its total energy using a simple one-step Newton method. Next we adjust the conditional probabilities for the node according to its new position and orientation. The total probability of the existence of the node is the sum of all the conditional probabilities obtained from the links with the other nodes. We perform this summation with a provision to avoid feedback loops and obtain the final probability or score of the node.

This iterative process of finding probabilities helps us make a better judgment of what should be included in the image graph with what score. For instance, a line segment that looks weak in the image is strengthened by having several strong connection with other nodes, all belonging e.g. to a truck. A strong looking line segment that is isolated from other nodes will end up having a low probability as part of any object and thus it will be considered noise.

3.7. Low-level Image Processing

A major hurdle for object recognition is reliable low-level image processing, e.g edge detection and line detection. How do we decide if a gradient is strong enough to be an edge? Or when is a curve straight enough to be a line? The field of computer vision has struggled with these questions since its inception. In fact there are no single answers to these questions - it depends on the context of the higher level objects. For instance, in a stick figure of a human, a stretched arm is made up of two distinct lines, a forearm and an upper arm, even if they are aligned in an almost straight line. However the forearm is really one line even though it is not perfectly straight. There is no way to figure this out until we have the context of a human body. In this project we deal with these issues using our dual-hierarchy graph. On the higher levels of our level of abstraction hierarchy we have quite loose requirements of objects, e.g. a curve is considered straight even if it is not perfectly straight. As we move down this hierarchy our standards become stricter and we consider the curve to be a line only if it is “really” straight. In this way we accommodate various options about lines and defer the decision whether something is a straight line or not until we gather more context from higher level knowledge.

Previously in this project we have used traditional methods of low-level image processing such as the Canny edge detector and connected components. This resulted in the usual amount of misdetected features and has limited the success of our detection. We have now applied our general hierarchy-based method to low-level image processing, as described above, to obtain much better results.

3.8. Learning and Modeling

We turn now to the question of how to build our graph representation from known data. As we have seen above, we have been able to use hierarchies to reduce the dimensionality of the problem. Specifically, we used the LOA hierarchy to recognize generic objects, setting aside the more specific information until we need it. We can use the same strategy for learning.

Our learning algorithm is analogous to the recognition algorithm described above: we go up in the parts hierarchy by grouping smaller parts, and we go down in the LOA hierarchy from generic to specific. Of course this time the graph is not known but needs to be built, either from examples or from explicit knowledge. For a starting point, we include in our graph a set of simple “innate” objects such as a circle and a 3D box. Such objects are not atomic, i.e. they are composed of simpler parts such as line segments, but they are still very generic, i.e. high in the LOA and low in the parts hierarchy. More specific or more complex objects are learned automatically from examples.

A simple example can illustrate this. Suppose we have already identified a certain object in the image as a “box” and some “circle” objects. These are very generic simple parts, at the top of the LOA and at the bottom of the parts hierarchy. They are close to each other in the image so we measure some generic relations that the system is programmed to examine. We discover that the box sits on top of the circles (a very common generic relation), so they may comprise a group. We do not yet have a node in the graph for this group so we create a temporary node. Looking at a different image, we can see a similar grouping. Before we create another temporary node we check if we already have a similar temporary node created from a previous image. If we do, the temporary node becomes more permanent. More images strengthen this node. This way we go up in the parts hierarchy.

We now try to go down in the LOA hierarchy and look for more specific relations between the parts in our new node. We discover that the circles are parallel to each other and to the side of the box (more accurately, the ellipses projected by the circles have parallel major axes with the same aspect ratio). We see that this relation is consistent across all our examples. This makes the node really strong and we call it a “vehicle”. Turning back to the parts hierarchy, we look for additional parts that may be associated with this vehicle. We discover that things like a steering wheel, a windshield, headlights etc. are usually associated with the vehicle so we add them to the list of parts. Turning back again to the LOA hierarchy we look for more specific relations among these parts. This is when we discover the difference between a car and a truck etc. In this way we alternate between going up in the parts hierarchy and down in the LOA hierarchy, assembling bigger groups and establishing more specific relations to create more specific models.

One can argue that biological vision systems have a certain number of innate generic objects too. For instance, the difference between specific predators such as a lion and a tiger can be learned from examples, but a generic predator, an animal with a big mouth that can eat you, is probably innate. Learning that from examples would be too late for the prey. Mythical monsters in movies or cartoons can look quite scary even though we have never seen their likes before. It is also possible that cars seem quite natural to us because they resemble some generic animal: they have a body, a front part with two shiny “head”-lights, a tail section and four extremities touching the ground. A learning algorithm is also innate. We cannot train a monkey to read beyond a few symbols.

3.9. Algorithm Implementation Details

The method has been implemented in MATLAB . The graph traversal algorithm, many geometric relations and basic shapes such as 3D boxes were tested on hundreds of real images and are fully operational. More shapes are being added. The learning algorithm still needs some work. This algorithm is typically run off-line and is not part of the recognition process.

In the following we briefly describe some details of the implementation.

Most modern programming languages such as MATLAB, C, Fortran95, have a data structure called a “structure array”. We arrange our graph representation as such an array. The structure contains fields that can be accessed by name and these can contain quite arbitrary content. In a typical example, the fields contain employee name, address, phone number etc. The fields typically have an index to account for different employees. In our implementation, the nodes in the graph are represented by such indexed fields. Each node has the following main fields, shown with the “truck” example:

Type: truck	The name ("type") of the object
Instance: 1	An object instance number
Parts: cabin, trunk-a, trunk-b, wheels	Names of part nodes including variants
Groups: convoy	Names of group nodes the object is linked to
Part(i) origin: *x,y,z*	Origin of part *i*
Part(i) axes: *ax1,ax2,ax3*	Axes directions (orientation)
Part(i) elasticity: *k(i)*	"spring" constant of part *i*
Higher LOA: vehicle	Names of nodes at higher LOA
Lower LOA: truck1, truck2	Names of nodes at lower LOA
Origin: *x,y,z*	Coordinates of object's origin
Axes: *ax1,ax2,ax3*	Directions of object's axes
Elasticity *k*	spring constant
Midx: cabin, trunk-a: truck1	Lower groups indexed by parts
Relations: ("ratio", width, cab , trunk-a, 2, 0.3)	Various relations between parts
("ratio", distance, cab , trunk-a, 1.5, 0.3)	indexed by higher parts

The data in the "relations" field is inspired by the syntax of Lisp. Each relation is a list whose first element is a name of a function, in this case "ratio". This function finds the ratio of sizes of part1 and part3, which are "cabin" and the variant "trunk-a" of the trunk (from the parts field). For this relation to be satisfied the function "ratio" has to return the value of "2" with tolerance 0.3. This function is one of several invariant relations functions that can be checked. They can be geometric such as "ratio" (checking ratios of sizes and/or distances) or "pose" (checking relative poses including parallelism), or they can be topological relations such as "touch" or "inside".

3.10. Improved SIFT-based Object Identification System

The second goal of our project was to identify an object which has already been detected, namely: given an image of a truck, search our database to find out if we have seen the same truck before. This "fingerprinting" system does not try to detect the truck (which is the first goal of the project) but relies for this on a human operator. This is useful for example when a soldier sees an image of a suspicious truck and tries to determine if someone has already archived an image of this particular truck before.

For this goal we have adapted a method that has already been proven useful in other contexts in the literature. It is based on the so-called SIFT features. We have adapted an off-the-shelf software package for an improved implementation of this method. The basic process consists of finding points of interest in the image, converting each such feature point into a 128-bit vector descriptor, performing vector quantization to reduce this vector set to a smaller "vocabulary" of "visual words", and then match these words between the query image and the dataset. This is done using histograms that sort these "words" according to their "popularity" in the images. Such histograms are built for both the database of known images and the unknown query image. If the histogram of the query image matches one in the database then the image has been identified.

One problem with this method in that the visual words are not always distinctive enough to identify an object. They become more distinctive if we use them in context, namely in "visual phrases". One recent method [3] accomplishes this by using random blocks. It can be summarized as follows:

- The data-set images are partitioned into overlapping blocks of random size and center.
- Each block contains a set of words, or a visual phrase. Each word (feature) appears in several phrases containing it.
- Each phrase (block) provides a vote in a confidence map for the presence and location of each feature in the image.

We have improved on this method by making it hierarchical and obtained a speed-up of 20 times, while reducing the number of false alarms. Instead of partitioning the image with random blocks many times, we use a hierarchy of blocks.

- Starting from large blocks at the highest level, intersect the histogram of the query image with that of each block. Choose the top 10 scoring blocks (the ones with biggest intersections) to go down to the smaller blocks right below them on a lower level.
- For each possible feature location, sum up the scores of all related blocks to obtain a confidence (voting) map, telling us the likelihood of having that feature in that particular location.

The blocks are illustrated in Figure 11. The image on the left shows various random blocks, and the graph on the right shows the word histogram of each block in the hierarchy.

The results are described in Section 4.3.

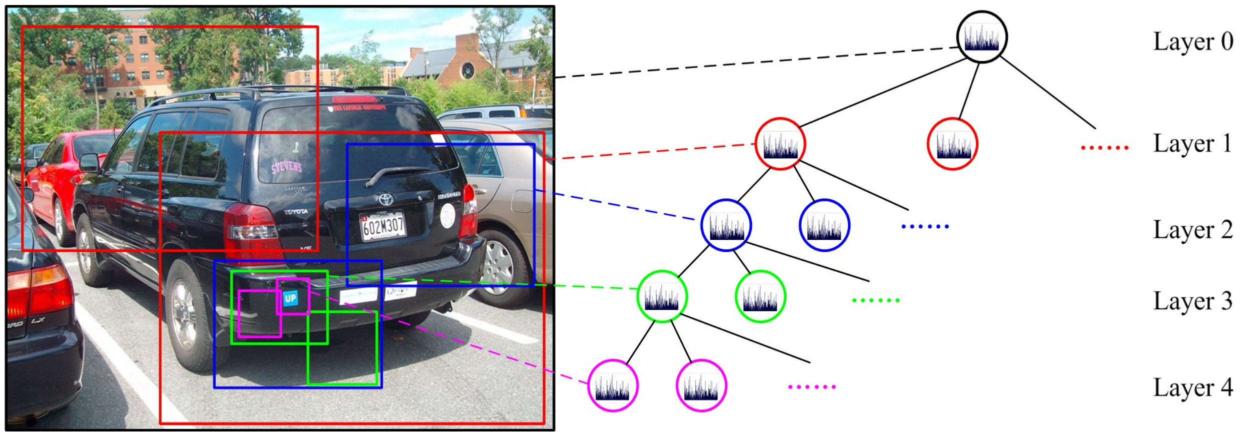

Figure 11: Hierarchical block structure

We can achieve further improvement by using an adaptive hierarchical spatial decomposition when we have some prior information about the vehicle before applying the decomposition. In this case we first run a vehicle detector on each database image with appropriate parameters to obtain bounding boxes containing only the vehicle. We used the common Felzenszwalb detector trained on the VOC2009 database. After obtaining the detection results, we perform the hierarchical spatial decomposition only around the detected bounding boxes.

Another improvement in performance was achieved by replacing the uniform number of sampled blocks with an adaptive sampling strategy. The number of child blocks is no longer uniform, at 25 blocks per parent, but depends on the number of SIFT features in the parent blocks. If we have more features we create more children for this parent blocks, while sparse features call for fewer blocks.

We have also addressed the issue of viewpoint invariance. Although standard SIFT descriptors are scale and in-plane rotation invariant, we need to compute more general 3D invariants which can be used to estimate the 3D structure and pose of the object.

A general method for dealing with the projection from 3D to 2D is described in Section 3.4, based on paper by I. Weiss in a PAMI paper in 2001 [10]. This method does not rely on modeling assumptions such as parallelism or symmetry as the graph method of Goal 1. While it does not recover arbitrary depth information, it places strict constraints on which 3D model can fit the projected image, which in practice it is usually sufficient. In that paper the descriptors were simply prominent points or vectors. In the current project we use more general descriptors: the so called affine Harris feature detector. This give us orientations of the "visual words" which are affine-invariant in 2D (only). We then apply our method mentioned above to these descriptors to derive the 3D structure and pose of the object. In effect, we replace the previous "spatial verification" of the SIFT method by invariant verification.

4. RESULTS AND DISCUSSION

4.1. Main Accomplishments

Our main accomplishments are as follows:

1. Built a general-purpose system for object detection and recognition, based on our dual-hierarchy graph and the graph traversal algorithm, going bottom-up in the parts hierarchy and top-down in the level of abstraction hierarchy.
2. Obtained viewpoint invariance and reconstruction of 3D objects from 2D images.
3. Improved on the common low-level processing such as edge and line detection.
4. Built a SIFT-based system for identification, or fingerprinting specific vehicles in a database.
5. Improved substantially SIFT method in reliability and speed by using hierarchical random visual phrases rather than visual words.

Our data set currently consists mainly of vehicles. These are man-made objects and are relatively easy to model. Other man-made objects such as weapons can be handled in a similar way.

As the method is based on intrinsic and invariant properties of objects and therefor it is robust to various external interferences. The ability to recognize generic objects on different levels of abstraction adds robustness with respect to variability, large or small, in the object itself. This is unlike current machine learning methods which require training for nearly every possible variation. Most often we do not have the specific model of a visible truck in the database. Even if we do, the specific truck is usually modified by adding or subtracting parts, by pose, shadows, painting, dirt, etc. One can never train a system for all possible variations.

4.2. Goal 1: Object Detection and Recognition

We focus here on the issue of viewpoint invariance and 3D reconstruction, which is one of the major advantages of our method.

A serious problem with current recognition methods is that they are not invariant to viewpoint changes. A different pose of the object can change the descriptors substantially and make recognition with common methods impossible. This is true for both the template-based methods normally used for object detection (our Goal 1) and the SIFT -based methods used for identification (our Goal 2). We have obtained viewpoint invariance in this project. Achieving viewpoint invariance also means that we can reconstruct the full 3D object from its projected 2D image.

In the most general case 3D reconstruction is an ill-posed problem, namely it does not have a unique solution, because the depth information is lost in the projection from a 3D object to a 2D image. However in practice we can make reasonable assumptions based on our knowledge of objects to find a reasonable solution. For example, man-made objects such as vehicles often posses properties such as symmetry, orthogonality of the main axes, parallelism of many visible lines etc. We incorporate these as part of our knowledge base of models. In our general recognition system we take advantage of these modeling assumptions, combined with the hierarchical structure, to match images such as in Figure 12. We recognize these as images of the same vehicle, in spite of the different viewpoints. In contrast, neither the current template-based recognition methods nor the SIFT-based methods can deal with viewpoint changes as they cannot incorporate geometric knowledge into the training.

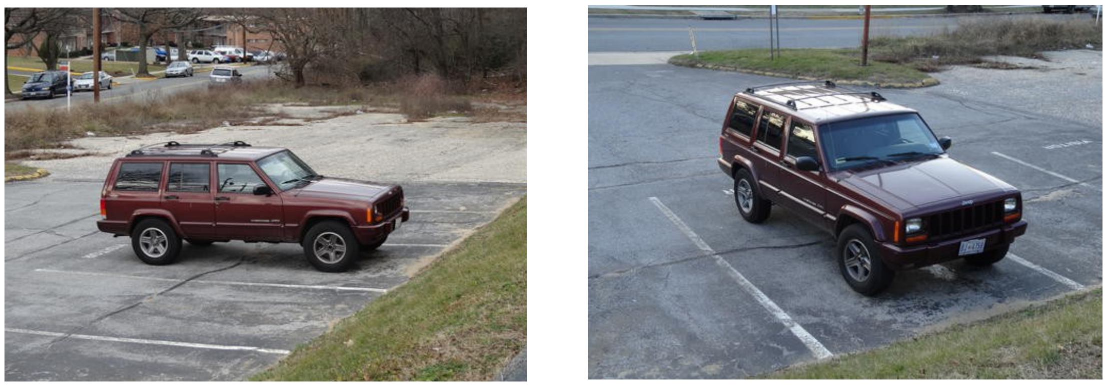
Figure 12: Our system recognizes these images as projections of the same 3D object.

Using the hierarchical graph method, the problem has turned out to be easier than expected. The method is model-based and part-based, so each part of a model has some information (in its own model) on how it may be projected from 3D to 2D. For example, we know that a parallelogram visible in 2D is likely to be a projection of a 3D rectangle, but we are missing the information of its tilt and slant in 3D. The hierarchical graph combines several such part models into a bigger model and informs us of the spatial relations among the parts, and that reduces the missing information. For example, obviously each rectangle in the bigger vehicle model cannot have its own independent tilt and slant - they all share the tilt and slant of the object as a whole. Thus this spatial relation reduces many unknown tilts to one. Combining all the relevant geometric information in the graph we end up with an unambiguous determination of the object's identity, namely the correct model that matches it. Given that, we can also find its position in 3D including its tilt and slant. Thus, a problem that has baffled the computer vision field since its beginnings, namely 3D reconstruction, has been solved here as a natural product of the method. Examples are shown in Figures 12-15.

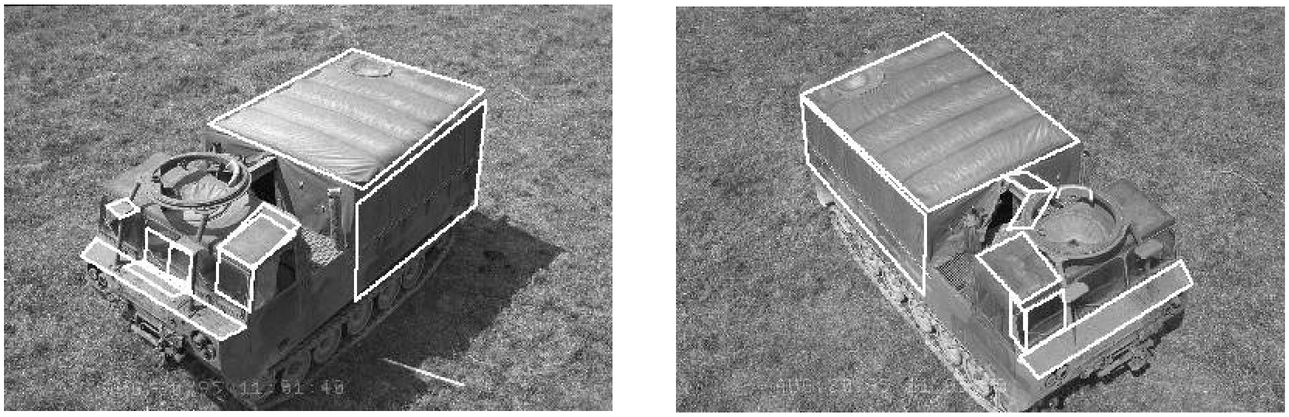

Figure 13: M54 truck, two views, similar ratios.

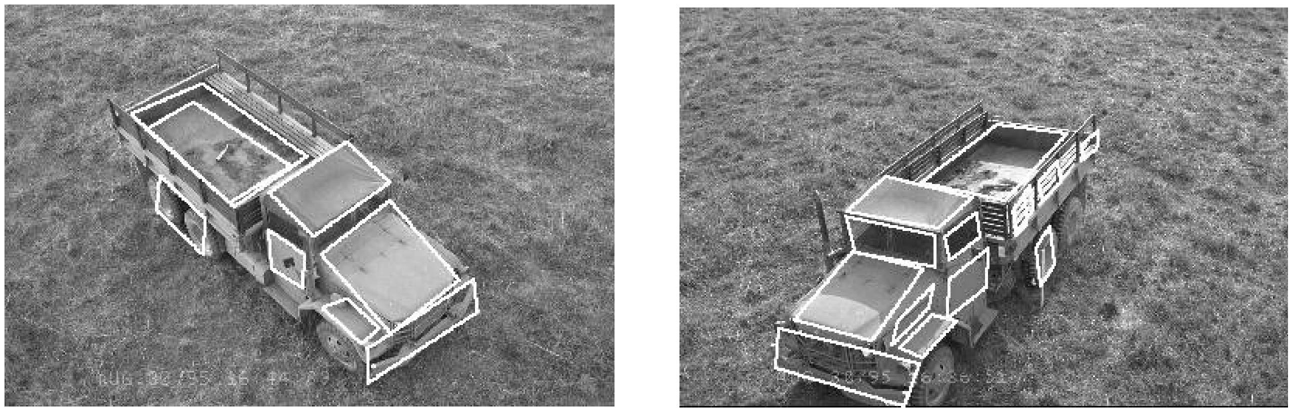

Figure 14: M35 truck, two views, similar ratios.

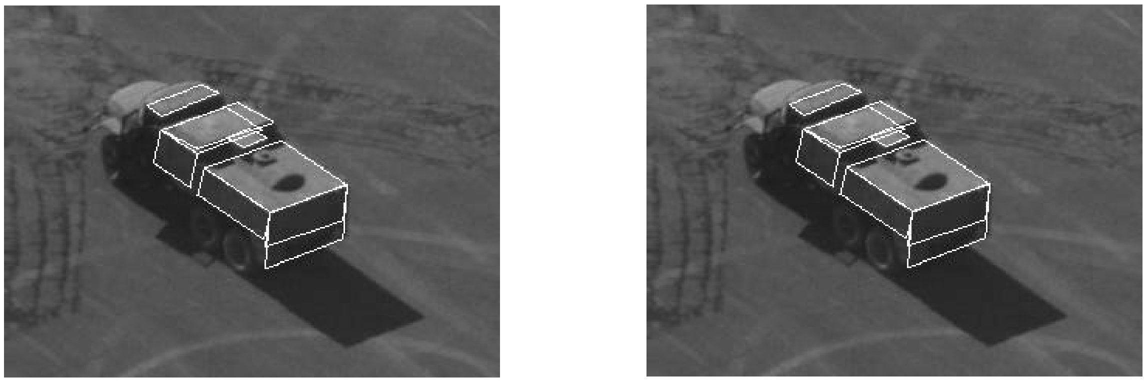

Figure 15: Truck, two views, similar ratios.

4.3. Goal 2: Object Identification

We have collected images of cars and built a database of about 400 images. We have successfully tested the method using this database. Our current implementation is based on C++ and is very efficient. It only takes on average around 100ms to process an image.

We have compared our method to the previous method described in Section 3.10.

In figure 16, the first image is the query and the second is the matching image that our system found in the database. The third image shows the confidence map of the previous, unimproved method. We can see a peak in the confidence, indicating a correct match. The fourth image is the confidence map of our improved method, indicating a correct match also.

Figure 17 shows that the previous method found a peak in the confidence map even though the pictures do not match. This is a false alarm. Our improved method does not show a peak in this case, avoiding the false alarm.

In addition to eliminating many false alarms our method achieved a speed-up of 20 times over the previous method..

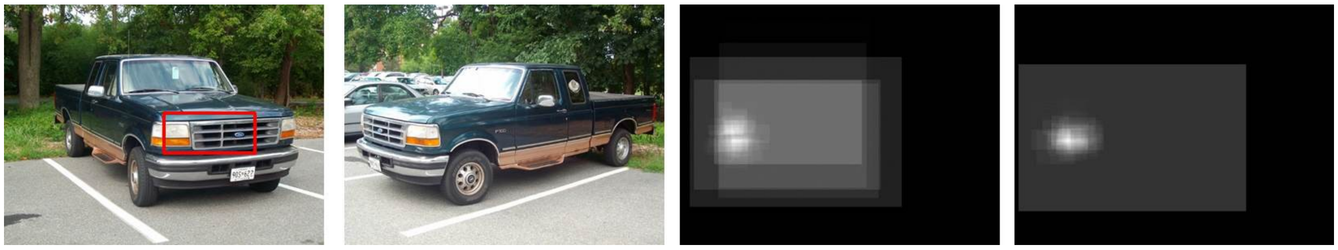

Figure 16: Correct match

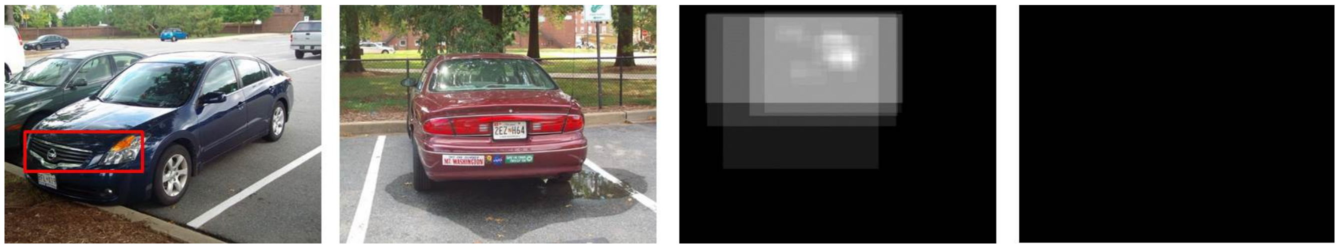
Figure 17: Incorrect match

5. CONCLUSION

An object recognition system described in Goal 1 is the holy grail of the ATR field and a successful development will dramatically increase the capabilities of any system that relies on automatic target recognition.

We have built a system consisting of the recognition algorithm, namely the dual hierarchy graph and the traversal algorithm, along with some basic models, in the form of a 7,000-line MATLAB code.

Our approach occupies the middle ground between two extremes: the “brute-force” machine learning approach, trying to learn everything from examples without much domain-specific knowledge, and the rule-based approach that relies on a large number of hard-wired rules without much learning. In our approach the basics are hard-wired and based on these the system can learn more from examples. This is made possible because our “rules” are not rigid but have the flexibility afforded by our graph structure.

As we discussed, we have to “earn our lunch” by applying domain specific knowledge, namely our understanding of images. Our specific knowledge consists of the following: the dual-hierarchy attributed graph structure, the algorithm of going up and down our hierarchies for recognition and learning, and an initial set of simple objects such as rectangles and 3D boxes.

A key innovation of our method is the use of a dual-hierarchy graph as opposed to previous methods which all used one-dimensional hierarchies of various kinds. Furthermore, we have developed an innovative method of traversing this graph. Previous methods used top-down or bottom-up strategies in a 1-D hierarchy. Our method combines the advantages and avoids the pitfalls of both. Our graph contains two interlocking hierarchies (Figure 1): (i) a parts hierarchy, e.g. a truck has a trunk and a cab, a car has a body, wheels, etc. (ii) a hierarchy describing a level of abstraction or how generic an object is, e.g. a vehicle is more abstract than a car. Our traversal algorithm progresses from the bottom up in the parts hierarchy and from the top down in the level of abstraction hierarchy.

Another key component is the use of geometrically invariant descriptors, making the system robust to viewpoint changes. This will be described later.

Current technologies are based on machine learning, which are in turn based on template matching. They use a large set of training templates and various methods of statistical inference to extrapolate from the training set to the object to be recognized. However it is well known that such systems cannot account for all the variations common in objects. The training set would be impractically large. Furthermore, such system cannot recognize a generic object such as a car, at best they can recognize a specific image of a specific car. Our system can detect and recognize generic objects in differing poses, illuminations etc.

The system is model-based, namely it relies of having a set of three dimensional models arranged according to our hierarchies. Simple common objects such as rectangles or 3D boxes have been coded manually. More complicated models can be learned from examples by a leaning algorithm that is analogous to our recognition algorithm.

6. REFERENCES

[1] Grenander, U. and Miller, M., [Pattern Theory: From Representation to Inference], Oxford University Press, (2007).

[2] Bishop, C.M., [Pattern Recognition and Machine Learning], Springer, (2006).

[3] Jiang, Yuning, Jingjing Meng, and Junsong Yuan, Randomized Visual Phrases for Object Search, IEEE Computer Vision and Pattern Recognition (CVPR'12), 2012.

[4] Schaffer, C., "A conservation law for generalization performance," International Conference on Machine Learning, H. Willian and W. Cohen, Editors. Morgan Kaufmann, 295-265 (1994).

[5] Wolpert, D.H. and Macready, W.G., "No free lunch theorems for optimization," IEEE Transactions on Evolutionary Computation, 67(1), (1997) .

[6] Weiss, I., "Projective invariants of shape," Proc. Computer Vision and Image Processing, 291-297 (1988).

[7] Weiss, I., "Noise resistant invariants of curves," IEEE Trans. Pattern Analysis and Machine Intelligence, 15(9), 943-948 (1993).

[8] Weiss, I., "Model-based recognition of 3D curves from one view," Journal of Mathematical Imaging and Vision, 10(2), 175-184 (1999).

[9] Rivlin, E. and Weiss, I., "Local invariants for recognition," IEEE Trans. Pattern Analysis and Machine Intelligence, 17(3), 226-238 (1995).

[10] Weiss, I. and Ray, M., "Model-based recognition of 3D objects from single images," IEEE Trans. Pattern Analysis and Machine Intelligence, 23(2), 116-128 (2001).